UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P O Box 1450
Alexandria, Virginia 22313-1450
www uspto gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/754,785	01/04/2001	Pierre-Alain Darlet	40101/06901	3238

30636 7590 04/29/2013
FAY KAPLUN & MARCIN, LLP
150 BROADWAY, SUITE 702
NEW YORK, NY 10038

EXAMINER	
RUTTEN, JAMES D	
ART UNIT	PAPER NUMBER
2197	
MAIL DATE	DELIVERY MODE
04/29/2013	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.** 09/754,785	**Applicant(s)** DARLET, PIERRE-ALAIN	
	Examiner James D. Rutten	**Art Unit** 2197	**AIA (First Inventor to File) Status** No

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *6 February 2013*.

□ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on _____.

2a)☒ This action is **FINAL**. 2b)□ This action is non-final.

3)□ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.

4)□ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

5)☒ Claim(s) *1-15 and 40-60* is/are pending in the application.

5a) Of the above claim(s) _____ is/are withdrawn from consideration.

6)□ Claim(s) _____ is/are allowed.

7)☒ Claim(s) *1-15 and 40-60* is/are rejected.

8)□ Claim(s) _____ is/are objected to.

9)□ Claim(s) _____ are subject to restriction and/or election requirement.

* If any claims have been determined allowable, you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov.

Application Papers

10)□ The specification is objected to by the Examiner.

11)□ The drawing(s) filed on _____ is/are: a)□ accepted or b)□ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

Priority under 35 U.S.C. § 119

12)□ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

Certified copies:

a)□ All b)□ Some * c)□ None of the:

1.□ Certified copies of the priority documents have been received.

2.□ Certified copies of the priority documents have been received in Application No. _____.

3.□ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Interim copies:

a)□ All b)□ Some c)□ None of the: Interim copies of the priority documents have been received.

Attachment(s)

1) □ Notice of References Cited (PTO-892)

2) □ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____.

3) □ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ .

4) □ Other: _____.

DETAILED ACTION

1. The reply filed February 6, 2013, has been received and entered. No claims have been amended or canceled. Claims 1-15 and 40-60 are pending and have been examined.

Response to Arguments

2. Applicant's arguments filed 2/6/13 have been fully considered but they are not persuasive.

3. On pages 2-4, Applicant essentially argues that prior art of record "Tool Interface Standard (TIS) Portable Formats Specification, Version 1.1" (hereinafter "TIS") fails to teach limitations related to "the reordering of the components is based on the section type for each of the components" as provided in claim 1. Specifically, Applicant argues that page 1-2, lines 6-8 of TIS teaches that "sections and segments have no specified order," so TIS fails to teach or suggest any ordering of the sections or segments. Initially, it is noted that prior art of record *Levine,* not TIS, is relied upon for disclosure of reordering. However, it is also noted that the rejection is based upon a broad but reasonable interpretation of the claimed limitations as informed by the disclosure of the specification as cited by the Applicant in the 10/9/12 response (see U.S. Patent Application Publication 2002/0087956 by Darlet, pars. [0034], [0046], [0058], and [0065]). Paragraph [0065] provides perhaps the most relevant support:

> [0065] In step 408, the section header table may be copied into the reordered module immediately following the program header table. Space may be allocated in the section header table to contain section header entries for each section in the software module. In the example embodiment, the number of sections is the number of text and data sections, and at least three fields are used in the section header table for each section. It will be appreciated that the procedure may be adapted if

> other sections, e.g., multiple text sections, or new types of section were added to the software module

Thus, the specification clearly envisions components which include section types, and that reordering occurs with components that may involve such section types. No particular description was found specifically describing a particular manner of reordering based on the section type. As such, no special interpretation is provided to the claimed limitations which appear to have been taught accordingly by TIS. Furthermore, even if TIS describes sections that "have no specified order" as argued by Applicant, that does not mean that TIS fails to teach the claimed elements. The claim requires reordering based on the section type. As informed by the specification, a component to be reordered based on section type merely requires that the component has a section type. TIS teaches components that include section types. Even if TIS fails to teach ordering, this does not diminish *Levine's* disclosure of reordering. Thus, TIS in combination with *Levine* appears to teach the limitation.

In response to applicant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986). In this case, prior art of record Levine teaches reordering modules to remove backward references as cited in the rejection. Secondary reference TIS is used to teach components that include section types. As noted in the rejection, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use Levine's component reordering with the section type and section header taught by TIS in order to conform to a known specification and streamline the software development process as suggested by TIS (see at least "Introduction" on page i). Thus, the rejection is based upon the combination of references, and not upon TIS alone.

At the bottom of page 3, Applicant essentially argues that the teaching of TIS does not permit the nonsequential reading required in claim 1. Similar to the reasoning provided above, *Levine*, not TIS, is relied upon for disclosure of this limitation. *Levine* discloses use of the ELF format taught by TIS. The relocation disclosed by Levine accommodates the shortcomings of TIS. It is the *combination* of *Levine* and TIS, and not TIS alone, which teaches the limitations.

Further arguments which have not been explicitly addressed are similar to the arguments addressed above, and are not persuasive for the same reasons.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-15, 40, 41, and 43-60 are rejected under 35 U.S.C. 103(a) as being unpatentable over John Levine, "Linkers and Loaders, chapter 6," June 1999 [online] accessed 08/15/2005, Retrieved from Internet <URL:

http://www.iecc.com/linker/linker06.txt>, 9 pages (hereinafter *Levine*) in view of prior art of record "Tool Interface Standard (TIS) Portable Formats Specification, Version 1.1" (hereinafter "TIS").

As per claim 1, *Levine* discloses receiving a software module, the software module including references to locations within the software module, at least some of the references being backward references; and reordering components of the software module into a predetermined order to remove at least some of the backward references, ... (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references), wherein the components further include at least one of a header, a section, and a table (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)).

Levine teaches the use of ELF libraries (see p. 2, e.g. "ELF libraries"). But *Levine* does not expressly disclose: wherein each of the components includes one of a plurality of section types and the reordering of the components is based on the section type for each of the components. However, TIS teaches the format for ELF libraries as including an indication of a section type. See TIS, pages 1-9 and 1-10, in particular Fig. 1-9, e.g. "Section Header," Fig. 1-10, e.g. "Section Types," and Fig. 1-14, "Special Sections." As indicated at the top of page 1-9, object file sections have a corresponding section header (including a section type) that describes it. In order to conform to the standardized ELF format, any section ordering would need to be "based on" a section type, since this is necessary information in a section header as taught by TIS. Note that the claimed reordering "based on the section type" is broadly but reasonable interpreted in light of the portions of the specification (published as U.S. Patent Application Publication 2002/0087956), cited by Applicant in the 10/9/12 response, including paragraphs [0034], [0046], [0058], and [0065]. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Levine's component reordering with the section type and section header taught by TIS in order to conform to a known specification and streamline the software development process as suggested by TIS (see at least "Introduction" on page i).

Levine further discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in some backward references remaining (see "Exercises" on p. 8). *Levine* further suggests a solution to remaining backward references, which includes listing the same library several times on the linker command line (see "Searching libraries" on p. 6-7), essentially replacing a backward reference with a forward reference to the repeated library entry, thereby loading (storing in memory) the components in such a manner as to avoid a nonsequential reading of the reordered software module, *i.e.*, the components are still read sequentially, but in a longer sequence that includes some repeats (for example, A B A, B A B, or A B C D A B C D, as described on p. 6 of *Levine*). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to utilize such storing of backward references in memory to avoid a nonsequential reading as a known means of handling a known limitation of attempting to remove such backward references.

As per claim 2, *Levine* further discloses adjusting at least one of the references in the software module to reflect the reordering of the components of the software module, so that the at least one of the references remains a reference to the same component, by to the component's new, reordered location, the new, reordered location coming after the at least one reference in the software module (see "Creating libraries" on pp. 5-6 and "Library formats" on pp. 1-5).

As per claims 3 and 4, *Levine* further discloses the software module including a symbol table, the symbol table including no backward references after the reordering and adjusting steps (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, *Levine* discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claims 5-8, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5). *Levine* further discloses the software module including a symbol table, the symbol table including no backward references after the reordering and adjusting steps (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, *Levine* discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claim 9, *Levine* discloses a reorder module configured to receive a software module including references to locations within the software module, at least some of the references being backward references, the reorder module configured to reorder components of the software module into a predetermined order to remove at least some of the backward references, (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references), the components further including at least one of a header, a section, and a table (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)). The use of a processor and memory is inherent in realizing the functionality of *Levine*.

Levine teaches the use of ELF libraries (see p. 2, e.g. "ELF libraries"). But *Levine* does not expressly disclose: wherein each of the components includes one of a plurality of section types and the reordering of the components is based on the section type for each of the components. However, TIS teaches the format for ELF libraries as including an indication of a section type. See TIS, pages 1-9 and 1-10, in particular Fig. 1-9, e.g. "Section Header," Fig. 1-10, e.g. "Section Types," and Fig. 1-14, "Special Sections." As indicated at the top of page 1-9, object file sections have a corresponding section header (including a section type) that describes it. In order to conform to the standardized ELF format, any section ordering would need to be "based on" a section type, since this is necessary information in a section header as taught by TIS. Note that the claimed reordering "based on the section type" is broadly but reasonable interpreted in light of the portions of the specification (published as U.S. Patent Application Publication 2002/0087956), cited by Applicant in the 10/9/12 response, including paragraphs [0034], [0046], [0058], and [0065]. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Levine's component reordering with the section type and section header taught by TIS in order to conform to a known specification and streamline the software development process as suggested by TIS (see at least "Introduction" on page i).

Levine further discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in some backward references remaining (see "Exercises" on p. 8). *Levine* further suggests a solution to remaining backward references, which includes listing the same library several times on the linker command line (see "Searching libraries" on p. 6-7), essentially replacing a backward reference with a forward reference to the repeated library entry, thereby loading (storing in memory) the components in such a manner as to avoid a nonsequential reading of the reordered software module, *i.e.*, the components are still read sequentially, but in a longer sequence that includes some repeats (for example, A B A, B A B, or A B C D A B C D, as described on p. 6 of *Levine*). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to utilize such storing of backward references in memory to avoid a nonsequential reading as a known means of handling a known limitation of attempting to remove such backward references.

As per claims 10, *Levine* further discloses adjusting at least one of the references in the software module to reflect the reordering of the components of the software module, so that the at least one of the references remains a reference to the same component, by to the component's new, reordered location, the new, reordered location coming after the at least one reference in the software module (see "Creating libraries" on pp. 5-6 and "Library formats" on pp. 1-5).

As per claims 11 and 12, *Levine* further discloses the software module including a symbol table, the symbol table including no backward references after the reordering and adjusting steps (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, *Levine* discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claims 13-15, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5). *Levine* further discloses the software module including a symbol table, the symbol table including no backward references after the reordering and adjusting steps (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, *Levine* discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claims 40 and 41, *Levine* further discloses linking the reordered module after the reordering (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claims 43-46, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

As per claims 47-54, *Levine* further discloses the reference pointing to/into a section or module before and after reordering (see "Creating libraries" on pp. 5-6 and "Library formats" on pp. 1-5).

As per claim 55, *Levine* discloses receiving a software module, the software module including components arranged in a first order, ... a first one of the components including a reference to a location in a second one of the components, the second one of the components preceding the first one of the components in the first order; and arranging the components into a predetermined second order so that the second one of the components is subsequent to the first one of the components in the second order, ... (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references), wherein the components further include at least one of a header, a section, and a table (see p. 2 (Libraries consist of an archive header, followed by alternating file headers and object files)).

Levine teaches the use of ELF libraries (see p. 2, e.g. "ELF libraries"). But *Levine* does not expressly disclose: wherein each of the components includes one of a plurality of section types and … wherein the arrangement is based on the section type for each of the first and second ones of the components. However, TIS teaches the format for ELF libraries as including an indication of a section type. See TIS, pages 1-9 and 1-10, in particular Fig. 1-9, e.g. "Section Header," Fig. 1-10, e.g. "Section Types," and Fig. 1-14, "Special Sections." As indicated at the top of page 1-9, object file sections have a corresponding section header (including a section type) that describes it. In order to conform to the standardized ELF format, any section ordering would need to be "based on" a section type, since this is necessary information in a section header as taught by TIS. Note that the claimed arrangement "based on the section type" is broadly but reasonable interpreted in light of the portions of the specification (published as U.S. Patent Application Publication 2002/0087956), cited by Applicant in the 10/9/12 response, including paragraphs [0034], [0046], [0058], and [0065]. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Levine's component reordering with the section type and section header taught by TIS in order to conform to a known specification and streamline the software development process as suggested by TIS (see at least "Introduction" on page i).

Levine further discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in some backward references remaining (see "Exercises" on p. 8). *Levine* further suggests a solution to remaining backward references, which includes listing the same library several times on the linker command line (see "Searching libraries" on p. 6-7), essentially replacing a backward reference with a forward reference to the repeated library entry, thereby loading (storing in memory) the components in such a manner as to avoid a nonsequential reading of the reordered software module, *i.e.*, the components are still read sequentially, but in a longer sequence that includes some repeats (for example, A B A, B A B, or A B C D A B C D, as described on p. 6 of *Levine*). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to utilize such storing of backward references in memory to avoid a nonsequential reading as a known means of handling a known limitation of attempting to remove such backward references.

As per claims 56 and 57, *Levine* further discloses linking the reordered module after the reordering (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claim 58-60, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

6. Claim 42 is rejected under 35 U.S.C. 103(a) as being unpatentable over *Levine* and TIS as applied to claim 1 above, and further in view of U.S. Patent No. 6,185,733 to Breslau et al.

As per claim 42, *Levine* discloses such a method but fails to expressly disclose transferring the reordered module to a different computer system and linking the module on the different computer system. However, *Breslau et al.* teaches the use of remote object libraries distributed prior to linking (see, for example, col. 4, lines 11-20). Therefore, it would have been obvious to one of ordinary skill in the computer art at the time the invention was made to such use of a different computer for linking. One would be motivated to do so, for example, to facilitate distributed software development efforts or reduce the physical storage requirements for object files (see, for example, col. 2, lines 4-25).

Conclusion

7. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action.

In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to James D. Rutten whose telephone number is (571)272-3703. The examiner can normally be reached on M-F 10:00-6:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Li B. Zhen can be reached on (571)272-3768. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/James D. Rutten/
Primary Examiner, Art Unit 2197